NLP for Stock Market Prediction with Reddit Data

Stanford CS224N Custom Project

Muxi Xu
Department of Computer Science
Stanford University
muxi@stanford.edu

Abstract

Reddit and the WallStreetBet subreddit has become a very hot topic on the capital market from the beginning of 2021. The discussions on these forums show the potential to influence the stock market. My project is to build a model to forecast the market movement based on the rich text data from Reddit. Specifically, I have explored sentence embedding, document embedding, CNN-based model, and sentiment analysis methods to leverage the sentence of posts & comments information for market forecasting. This project has tested and compared several types of model architectures. So far, the performance shows that the model could slightly improve performance from the naive forecasting method.

1 Key Information to include

- External collaborators (if you have any): None
- External mentor (if you have any): None
- Sharing project: None

2 Introduction

Recently, the Reddit forum has become very popular in the capital market. The subreddit WallStreetBets is believed to initiate a short squeeze on GameStop, which lead to a sharp increase in its price. Such a short-squeeze caused loss topping US$70 billion for some hedge funds in a few days. While staying unnoticed in the capital market previously, Reddit has drawn a lot of attention after these events. In this project, I seek the answer the following questions: Does the text in this Reddit have the information to predict the market movement? To this end, I have collected the text of this Reddit from Jan/2020 till Feb/2021, and then use the previous trading day's test to predict the current day's market movement.

To achieve this task, one major challenge is that the text gathered is lengthy. The data I gathered has a total of 2,623,978 sentences or 22,868,374 words. Meanwhile, if we build a model on the daily frequency, we only have around two hundred trading days' information. Due to this imbalance of dependent and independent variables, a model with appropriate complexity is the key to a successful prediction. A model with too many parameters may lead to overfitting, while a model with too few parameters may not able to capture the information in the data.

My approach could be summarized as follows: 1) Data are collected from two Reddit APIs, and pre-cleaned 2) the raw data is then processed to generate numeric measures with a) Sentence Embedding based on BERT[1] [2] b) Each Reddit Posts and following comments are combined as a document, which is embedding with the Doc2Vec Algorithm introduced by Le and Mikolov [3]. c) a daily sentiment has also been obtained by leveraging TextBlob and VADAR [4]. 3) then a CNN-based model is developed to predict the stock movement.

3 Related Work

There are many papers on extracting information from sentences and documents. Sanjeev, Yingyu Liang, and Tengyu [5] has proposed a method to use a weighted average of work vectors to generate the sentence representations. Zhe, Yunchen, Ricardo Henao, et al. [6] has developed a model with CNN and LSTM to encode sentences. Ryan, Yukun, Ruslan, et al. [7] uses the continuity of text to train an encoder-decoder model and then uses the vector obtained to represent the sentence. BERT has also been used in sentence representations. Sentence-BERT [2] provides a modification of the pre-trained BERT network that uses siamese and triplet network structure to derive semantically meaningful sentence embeddings. Furthermore, Le and Tomas [3] introduces a paragraph vector that enables the embedding of various length text like document or paragraph.

There is also a lot of work on financial forecasting with the NLP method. Frank, Erik, and Roy [8] has done a review of recent papers in the field of natural language-based financial forecasting. This paper has reviewed the works in this field from the 1980s, and it also introduces some developments in using social media contents for forecasting. In the field of leveraging social media content, many papers focus on using Twitter data because of its simple semantics and restricted length. Machine learning methodologies have also been widely used in this field. Xioa, Yue, et al.[9] use the neural tensor network and convolutional network to forecast S&P 500 index based on news text. [10] has used the sentiment score on Twitter data to predict stock market movement.

Some previous projects of CS230N also explored the area of financial forecasting with the NLP method. [11] has used sentiment embedding and document embedding to predict FRB rate changes from the FOMC meeting text. [12] uses the document embedding method to predict the well-known Fama French equity market factors. [13] uses an attention-based model on the company's 8k report to predict stock movement post-earnings.

However, the Reddit event happens recently and perhaps is still developing. To the extend of my knowledge, there is not much research on building NLP models on these data, which is the goal of my project.

4 Approach

4.1 Data Processing

Data processing is often an overlooked, but very important part of modeling. To facilitate the model development, I have done the following data processing steps:

- **data pre-clean**: with the data scrabbed from the Reddit API. I first did some preliminary data cleaning such as remove links, special symbols, etc.
- **Ticker Data Extracting**: with the pre-cleaned data, I apply a simple regular expression to extract all the ticker mentioned within each day.
- **Sentenece dataset**: Still on the preliminary cleaned data, I use the nltk package to split all the comments, posts, etc., into sentences.
- **Posts document dataset**: With the sentence dataset developed, I further merged those text under the same post together as a document. A subtlety here is we need to filter those comments based on timestamps to avoid any information leakage. For example, if any of today's comments on yesterday's post is mistakenly included, we may end up using future information for prediction.
- **Model Target**: Regarding the model target, I set my target as the market movement which is calculated as $Movement_t = I(Open_t - Close_t)$.

4.2 Sentence Embedding

With the preliminary datasets created above, I have used the sentence transformer python package to get the sentence embedding based on the BERT method. [2] [1]. Based on the paper [2] , this SBERT has added pooling operations to derive the fixed size sentence embedding. To fine-tune the BERT model, the authors have also created siamese and triplet networks to update the weights. After this sentence embedding process, a vector with a size of 768 dimensions is created for each sentence.

4.3 Document Embedding

While sentence embedding is created, naturally I also explored the document embeddings. To achieve a document level embedding. I have explored the following methods:

- **CNN Model on Sentence Sampling**: With the sentence embedding vector generated, a very straightforward way is to use a neural network to generate a vector that represents the information of a day. However, as described earlier, the number of sentences is different from day to day. To solve this problem, I've tried to sample 500 sentences from each day's post and use this as the input for the CNN model.
- **Average of all Sentence**: Another simple method I have tried is to take the simple average of all the sentence vectors for each day.
- **Doc to Vector on Posts documents**: A third method I have tried is to use the document to vector method introduced by [3]. As described earlier, this method is developed based on a document vector built in the model. For the document embedding, I have fit a model on the training data to obtain a vector of size 50.

4.4 Sentiment Analysis

Inspired by [10], I have also tried to include those common sentiment analyses into my model. Two sentiment analysis has been tried, the Textblob and VADER.

4.5 Neural Network Model

With all the above embedding and sentiment scores. I have developed couple of different neural networks to forecast the final market movement. My models have four major components or modules as in figure 1.

- **Data Sampling**: As described earlier, the length of documents varies from day to day. To get a fixed length of data for model training. I have tried to sample a fixed number of sentences or documents for each day. Currently, I have sampled 500 sentences and 10 documents for each day. In addition, my early model result shows a serious problem of overfitting. Thus, I have also tried to do mulitple sampling for each trading day. My current model is fitted on 5 samples from each trading day.
- **Setnence CNN Module**: This module takes the embedding vectors of the sentence and feeds them to a CNN model. The ideal is to leverage the CNN model to extract the information from embedding vectors.
- **Sentence Averaging Moudule**: The averaging module using simple averaging to combine all sentence vectors.
- **Document Module**: This module refers to using document embeddings instead of sentence embedding for forecasting. Similar to the sentence embedding, a CNN model is used to generate the forecast.
- **Sentiment Analysis**: This module takes the text of each day to generate a daily sentiment score time series.

With the above modules, I have built several models:

- **Sentence CNN Model**: This model uses a sentence CNN module and an FC layer to forecast the market movement.
- **Sentence Averaging Model**: This model uses a sentence CNN module and an FC layer to forecast the market movement.
- **Sentence CNN + Sentiment Model**: This model uses sentence CNN as well but before the FC layer. The sentiment score is added as well.
- **Document CNN + Sentiment Module**: This model uses document CNN and adds the sentiment score right before the FC layer.
- **Sentence + Document CNN + Sentiment Module**: This model uses almost all modules including sentence CNN, document CNN, and sentiment scores.

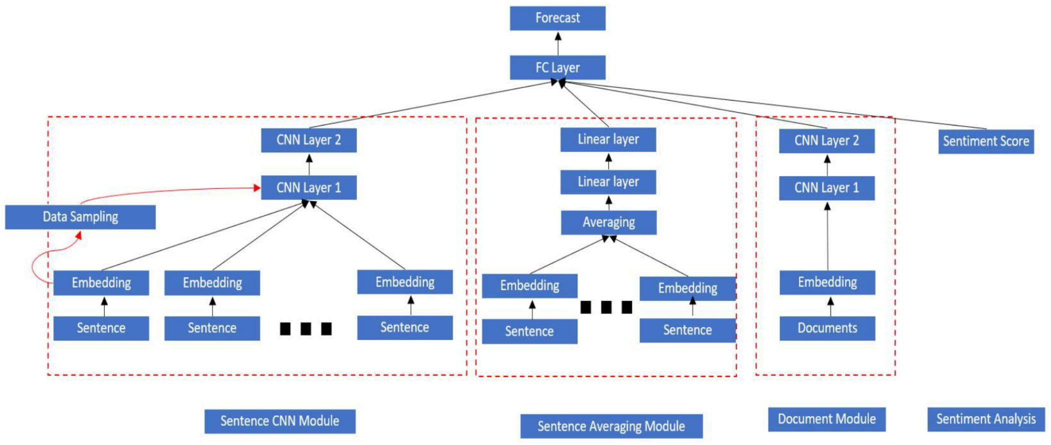

Figure 1: Model Structure

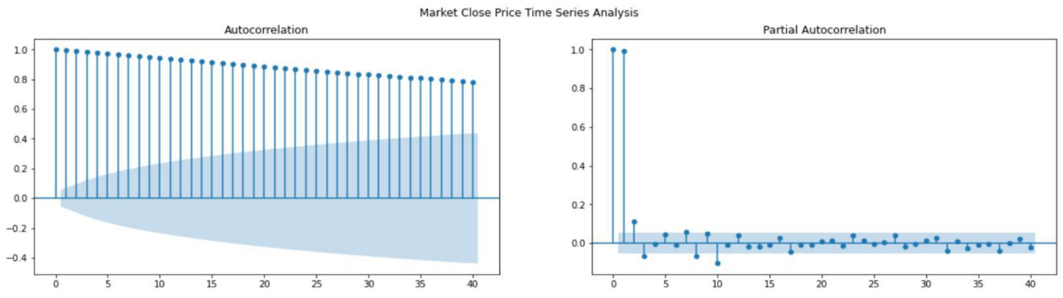

Figure 2: Closing Price Time Series Analysis

4.6 Baseline

For the baseline, I use the the most straightforward naive forecast method. As introduced in this article [14], the most simple way is to use current market price or condition to forecast future. In my case, the forecasting formula is:

$$Movement_t = Movement_{t-1}$$

This sounds simple but works. As shown in the figures 2 and 3, the stock market price shows a strong correlation with historical performance.

As we could see from the ACF/PACF analysis, the market price dataset shows a very strong autocorrelation pattern. While more complicated may be built, a simple AR1 method usually could generate a fair forecast.

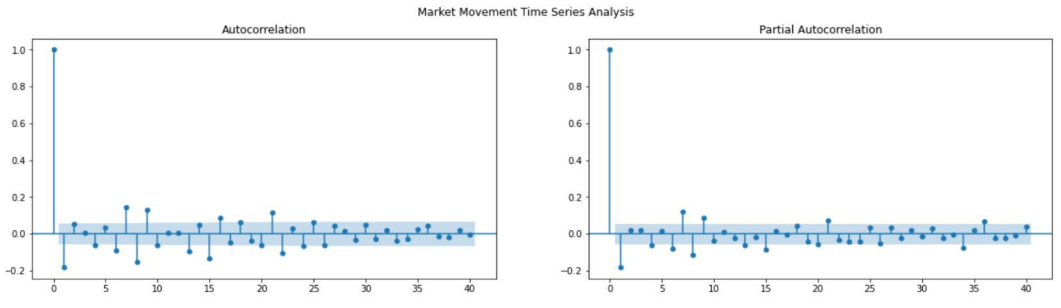

Figure 3: Market Daily Movement Time Series Analysis

5 Experiments

5.1 Data

My project needs two major data, the Reddit text data and the market price data.

- **Reddit Data**: Fortunately, there are a lot of materials about grabbing data from Reddit. [] I have modified those code to grab the data from wallstreetbets. As a quick summary:
  - **Time Range**: Jan, 2020 to Feb, 2021
  - **Nubmer of Sentence**: 2,623,978
  - **Nubmer of unique word**: 427,160
  - **Maximum/median/ minimum length of sentence**: 8,693/7/1
- **Market Data**: SP500 index price data is downloaded from Yahoo Finance. This data contains couple key columns including open price (OPEN), close price (CLOSE), max price of the day (HIGH), min price of the day (LOW), volume of the day (VOLUME).

5.2 Data Split & Evaluation method

My model is built on time series data. A random split of train & test dataset may not be appropriate. Thus, I split my data to in and out of time. Specifically, I use all the data in 2020 as training and after 2020 for testing. As the goal of my model is to forecast the movement of the market (up or down), I just use the accuracy ratio to evaluate the model, which is calculated as follows:

$$accuracy = \sum(I(Y_{fcst} == Y_{actual}))/N$$

5.3 Experiment Details

My experiement setup is slightly different among models. But in generally, one typical CNN module contains one Conv2d layer, one Norm2d layer, one ReLU activation, one Max Pool and one drop out layer. For my Doc2Vec or Sentence2Vec model, it may contain one or two such modules. For sentence average method, it only contains simple linear layers. The list belows shows some key items of my experiments:

- **training/Testing Period**: Jan, 2020 to Dec 2020/Feb, 2021
- **Learning Rate**: starts with 0.006 and deacy as more epochs.
- **Nubmer of Epoch**: 400 to 650
- **training time**: around 50 mins to 2 hours
- **CNN layer type 1**: kernel size: 3, stride 1, padding 1, input channel 1, output channel 4, dropout 0.2 to 0.4
- **CNN layer type 2**: kernel size: 3, stride 1, padding 1 , input channel 4, output channel 1, dropout 0.2 to 0.4
- **Linear Layer**: 32 or 64 hidden nuerons.

5.4 Experiment Results

As a result of the experiments, table 1 shows the accruacy of in and out sample test. Some observation from the experiment result.

- Model is able to improve the in-sample accuracy significantly. However, the improvement of out sample accuracy is not very significant.
- Model is prone to over fitting problem. As we could see, for those relatively more complicated models, the in-sample accuracy is all above 90%.
- Among all the models tested, the sentence embedding average model shows the best performance on the over-fitting aspect. Compared with other models, the difference between in-sample and out-sample accuracy is the smallest. I believe this model could be further improved.

Table 1: Model Results

#	Method	Training Period	Testing Period
0	Baseline	47.9%	46%
1	Sentence Embedding Averaging Model	61%	51.4%
2	Sentence Embedding CNN Model	100%	57%
3	Document Embedding CNN Model	91%	54%
4	Sentence Embedding CNN + Sentiment Model	93%	30%
5	Document Embedding CNN + Sentiment Model	83%	46%
6	Document & Sentence Embedding CNN + Sentiment Model	98%	51%

- The overfitting problems shows that we may need more data to train complicated model. One possible solution is to forecast single stock performance instead of the general market. For this, I have done the analysis of the ticker frequency. However, I haven't had enough time to finish the modeling on single stock level.

6 Conclusion

In conclusion, the models based NLP method does provide better prediction than the naive forecasting method. My result also shows sentence embedding CNN model has the best out of sample performance. However, from the overfitting aspect, the more simpler averaging model shows the better potential for further improvement. During the project, there are couple items I have tried but don't have enough time to finish.

- **Single Stock Forecasting**: a more reasonable assumptions is that the discussion on Redidt may have a larger influence on specific stock like GME. I have extracted the ticker frequency from each day's text data. We probably could build some models on these stocks' performance.
- **Higher Frequency**: my current model is done on daily frequency. As we have the over fitting problem, it may worth to try higher frequency like hourly.

References

[1] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics.

[2] Nils Reimers. Sentence transformers: Multilingual sentence embeddings using bert / roberta / xlm-roberta & co. with pytorch. In *https://github.com/UKPLab/sentence-transformers*, March 2021.

[3] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents, 2014.

[4] Clayton J. Hutto and Eric Gilbert. In *ICWSM*.

[5] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. A simple but tough-to-beat baseline for sentence embeddings. In *ICLR*, 2017.

[6] Zhe Gan, Yunchen Pu, Ricardo Henao, Chunyuan Li, Xiaodong He, and Lawrence Carin. Unsupervised learning of sentence representations using convolutional neural networks. 2016.

[7] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Skip-thought vectors. *CoRR*, abs/1506.06726, 2015.

[8] Frank Z. Xing, E. Cambria, and R. Welsch. Natural language based financial forecasting: a survey. *Artificial Intelligence Review*, 50:49–73, 2017.

[9] Pisut Oncharoen and Peerapon Vateekul. Deep learning for stock market prediction using event embedding and technical indicators. pages 19–24, 08 2018.

[10] Johan Bollen, Huina Mao, and Xiaojun Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8, 2011.

[11] Ye Ye. Can fomc minutes predict the federal funds rate? 2020.

[12] Andrew Han. Financial news in predicting investment themes. 2020.

[13] Mohamed Masoud. Attention-based stock price movement prediction using 8-k filings. 2014.

[14] Train2Test. Statistical approach to stock price prediction: Naive forecast, moving average. 2020.